Facultad 3

Aplicación móvil iLex Notario 2.0

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Frey Fernando Griñan Despaigne
Tutor(es): P.A., Ing. José Miguel Fabra Gallo, Ms.C.
Ing. Felix Antonio Marrero Pentón
Ing. Odetta Rey Garcia

2023

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Frey Fernando Griñan Despaigne soy el autor del Trabajo de Diploma: “Aplicación móvil iLex Notario 2.0.” y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración de autoría en La Habana a los 21 días del mes de noviembre del año 2023.

Frey Fernando Griñan Despaigne

_____
Firma del Autor

MSc. José Miguel Fabra Gallo

_____
Firma del Tutor

Ing. Odetta Rey Garcia

_____
Firma del Tutor

Ing. Felix Antonio Marrero Pentón

_____
Firma del Tutor

DATOS DE CONTACTO

MSc. José Miguel Fabra Gallo: Graduado de Ingeniería en Ciencias Informáticas en el año 2014 en la Universidad de las Ciencias Informáticas (UCI). Entre 2014 y 2018 se desempeñó como profesor de la disciplina Sistemas Digitales y como Asesor Docente-Metodológico del Vicedecanato de Formación de la Facultad 3. En el Centro de Gobierno Electrónico (CEGEL) ha desempeñado varias responsabilidades, entre ellas: jefe de departamento de desarrollo de aplicaciones, metodólogo y líder de proyectos de I+D+i. Posee experiencia en la tutoría de estudiantes de pregrado.

Correo electrónico: jmfabra@uci.cu

Ing. Felix Antonio Marrero Pentón: Graduado de Ingeniería en Ciencias Informáticas en el año 2017 en la Universidad de las Ciencias Informáticas (UCI). Desde su ingreso al Centro de Gobierno Electrónico ha trabajado en el proyecto LexCuba siendo desarrollador de las aplicaciones móviles iLex Constitución, iLex Leyes, iLex Reforma, iLex Notario, iLex MINJUS. Posee experiencia en la tutoría de estudiantes de pregrado.

Correo electrónico: famarrero@uci.cu

Ing. Odetta Rey García: Graduada de Ingeniería en Ciencias Informáticas en el año 2022 en la Universidad de las Ciencias Informáticas (UCI). Desde su ingreso en el Centro de Gobierno Electrónico (CEGEL) se ha desempeñado en el rol de Analista, trabajando en proyectos como Civix, ilexCuba y Conxul.

Correo electrónico: odettarg@uci.cu

AGRADECIMIENTOS

Primero que todo quiero darle gracias a mi madre por aguantarme y por estar siempre desde el inicio de este difícil camino, a mi padre donde quiera que este por ser mi pilar para aguantar los momentos más duros de esta hermosa carrera.

A mi hermano Eliober por ser mi segundo padre y guía para lograr estos resultados. A toda mi familia en general que me apoyaron en todo momento.

A mi novia Adriana por estar cada día a mi lado dándome fuerza.

A mis tutores por su paciencia, apoyo y dedicación en la elaboración de esta investigación.

A Emilio y Fernando por toda la ayuda que me brindaron durante el desarrollo de la tesis.

A mi familia de amigos con los que viví los años más lindos de mi vida, María, Isbel, Ernesto, Brayan…, y otros con los cuales me disculpo por no mencionarlos, a todas esas personas que de una forma u otra me acompañaron en toda esta aventura llamada UCI muchas gracias.

DEDICATORIA

Esta tesis está dedicada a mis padres que nunca dejaron que me diera por vencido y a mi hermano Eliober por ser mi ejemplo de superación.

RESUMEN

El Ministerio de Justicia de la República de Cuba (MINJUS) es uno de los órganos con resultados positivos en el proceso de informatización de la sociedad; como resultado de una estrategia para la informatización de sus actividades y áreas de trabajo desde los servicios magistrales y notariales hasta la gestión interna, procesos y actividades; un ejemplo de lo antes mencionado es iLex Notario, desarrollada de conjunto con el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI) para facilitar el acceso de la población a información notarial en el país. En la actualidad la aplicación móvil presenta un conjunto de carencias, que afectan la calidad de la información publicada a través de ella, así como el modo en que los usuarios pueden adaptarla a sus necesidades: entre las más significativas se encuentran la información desactualizada y que no cuenta con opciones para la configuración y ajuste de la apariencia. En consecuencia, con lo antes descrito, el presente trabajo de diploma tiene como objetivo principal: desarrollar la versión 2.0 de la aplicación móvil iLex Notario de manera que facilite la consulta de información actualizada sobre las notarías en Cuba. Para la realización de la propuesta solución se empleó como metodología de desarrollo de software: AUP (Proceso Unificado Ágil) en su variación UCI. Se utilizaron las herramientas, lenguajes y tecnologías definidas por el equipo de arquitectura del proyecto. Con el objetivo de garantizar la calidad de la aplicación se realizaron las validaciones y pruebas pertinentes.

PALABRAS CLAVES

aplicación móvil, dispositivo móvil, función notarial, notarias, servicios notariales

ABSTRACT

The Ministry of Justice of the Republic of Cuba (MINJUS) is one of the bodies with positive results in the process of computerization of society; as a result of a strategy for the computerization of its activities and work areas from magisterial and notarial services to internal management, processes and activities; An example of the aforementioned is iLex Notario, developed in conjunction with the Electronic Government Center (CEGEL) of the University of Informatics Sciences (UCI) to facilitate the population's access to notarial information in the country. Currently, the mobile application has a set of shortcomings that affect the quality of the information published through it, as well as the way in which users can adapt it to their needs: among the most significant are outdated information and It does not have options for configuration and adjustment of appearance. Consequently, with the above described, the main objective of this diploma work is to develop version 2.0 of the iLex Notario mobile application in a way that facilitates the consultation of updated information about notaries in Cuba. To carry out the proposed solution, the following software development methodology was used: AUP (Agile Unified Process) in its UCI variation. The tools, languages and technologies defined by the project architecture team were used. With the aim of guaranteeing the quality of the application, the relevant validations and tests were carried out.

KEYWORDS

mobile app, mobile device, notarial function, notaries, notary services

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha generado cambios en prácticamente todas las áreas y estructuras de la sociedad moderna; incluyendo las entidades jurídicas y de gobierno, las que deben actuar en nuevos escenarios donde la interacción entre los individuos está fuertemente mediada por las tecnologías y el acceso a la información es cada vez más necesario. En este contexto, las TIC se posicionan no sólo como un instrumento para mejorar el desempeño de este tipo de instituciones, sino que contribuyen a forjar nuevos modelos de gobierno basados en la gobernanza inteligente, el gobierno abierto y la innovación colaborativa. Además, ponen a disposición de los ciudadanos herramientas que le permiten incrementar su cultura jurídica (Eguía Peña, 2002).

Ante estas nuevas realidades, Cuba desarrolla un proceso de informatización de la sociedad, encaminado a la aplicación ordenada y masiva de las tecnologías en la gestión de la información y el conocimiento, para satisfacer gradualmente las necesidades sociales (…) lograr más eficacia y eficiencia en los procesos (…) y el aumento de la calidad de vida de los ciudadanos (Ministerio de las Comunicaciones de la República de Cuba [MINCOM], 2018). Al respecto, durante una reunión de la Asamblea Nacional del Poder Popular, el presidente de la República de Cuba Miguel Díaz-Canel Bermúdez ha planteado que existe una voluntad política de apoyar el proceso de informatización y que es preciso potenciar la soberanía, con software hechos en el país (Díaz-Canel, 2018).

El Ministerio de Justicia (MINJUS) es el organismo de la Administración Central del Estado encargado de asistir en materia legal al Estado y al Gobierno, así como proponer y, una vez aprobadas, dirigir las políticas que en materia jurídica le correspondan (Ministerio de Justicia de la República de Cuba [MINJUS], 2023).

Es uno de los órganos con resultados positivos en la informatización de sus actividades y áreas de trabajo desde los servicios magistrales y notariales hasta la gestión interna, procesos y actividades (MINJUS, 2023). Ejemplos de lo antes mencionado son: la Plataforma Bienestar para la solicitud y expedición de certificaciones del registro del estado civil; los portales del MINJUS y de la Gaceta Oficial de la República de Cuba; la digitalización de los registros del estado civil; aplicaciones móviles para los ciudadanos como iLex MINJUS, iLex Asesoramiento Jurídico e iLex Notario. Esta última, es el resultado de una colaboración entre el Centro de

Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI) y la Dirección de Notarías del MINJUS para facilitar el acceso de la población a información notarial en el país.

La referida aplicación móvil ha recibido valoraciones positivas por parte de la población y directivos del MINJUS, sin embargo, en la actualidad se pueden apreciar aspectos que afectan la calidad de la información publicada a través de ella, así como el modo en que los usuarios pueden adaptarla a sus necesidades:

- Las secciones marco legal, directorio de unidades y de unidades notariales se encuentran desactualizadas, provocando molestias en las entidades notariales correspondientes y confusiones en los usuarios a la hora de realizar los trámites que en la aplicación se mencionan.
- La aplicación móvil no cuenta con opciones para la configuración y ajuste de la apariencia; dígase colores, tamaño del texto y tema. Elemento que limita a las personas con dificultades visuales acceder al contenido.

Los elementos antes mencionados limitan el impacto positivo de esta solución informática en la ciudadanía, por lo que se define el siguiente **problema a resolver**: ¿Cómo perfeccionar la aplicación móvil iLex Notario de manera que facilite la consulta de información actualizada sobre las notarías en Cuba?

Para dar solución al problema anteriormente planteado se define como **objetivo general**: Desarrollar la versión 2.0 de la aplicación móvil iLex Notario de manera que facilite la consulta de información actualizada sobre las notarías en Cuba. Se determina el siguiente **objeto de estudio**: Desarrollo de aplicaciones móviles.

Delimitando como **campo de acción**: Desarrollo de aplicaciones móviles con información notarial.

Para dar cumplimiento al objetivo general propuesto se definen los siguientes **objetivos específicos**:

- Sistematizar los referentes teórico-metodológicos en los que se sustenta la propuesta de solución relacionados con el desarrollo de aplicaciones móviles con información notarial.

- Realizar la identificación de los requisitos, análisis, diseño e implementación de la aplicación móvil.
- Validar el diseño y el funcionamiento de la aplicación propuesta, aplicando métricas y pruebas de software respectivamente.

Como **idea a defender** de la investigación se plantea que el desarrollo de la versión 2.0 de la aplicación móvil iLex Notario, facilitará la consulta de información actualizada sobre las notarías en Cuba.

Los métodos de investigación se clasifican en teóricos y en empíricos. Los métodos teóricos de investigación implican la utilización del pensamiento y el razonamiento para ejecutar deducciones, análisis y síntesis. Por otro lado, los métodos empíricos de investigación se aproximan al conocimiento mediante experiencias replicables, controladas y documentadas, que se conocen bajo el nombre de experimentos (Andrés, 2017).

Con el propósito de resolver el problema y lograr el objetivo planteado, se emplearon los siguientes **métodos de investigación:**

Métodos teóricos

- **Análisis histórico-lógico:** se aplicó este método para establecer el estado del tema de investigación. Se estudió la existencia de aplicaciones similares o relacionadas con el tema en cuestión y su comportamiento en el decurso del tiempo, lo cual permitió profundizar en sus particularidades y sacar experiencias propias para la futura aplicación.
- **Analítico-sintético:** el método se utilizó para analizar la documentación referente al tema de investigación, lo cual permitió la extracción de los elementos más importantes que se relacionan con el objeto de estudio.
- **Modelación:** permitió un mejor entendimiento del proceso a informatizar. Contribuyó a modelar los artefactos esenciales para el desarrollo de la aplicación móvil.

Métodos empíricos:

- **Entrevista:** se utilizó en el intercambio con el cliente para adquirir información sobre el negocio y los requisitos que se deben cumplir en el desarrollo de la aplicación móvil. La guía empleada para la entrevista puede consultarse en el (Anexo 1).

Estructura del documento:

Capítulo I. Referentes teórico-metodológicos sobre el objeto de estudio: Se da a conocer los referentes teóricos de la investigación. Se realiza un estudio del estado del arte de las tecnologías ya existentes que tienen relación con el problema que se aborda. Se describe la metodología de desarrollo utilizada, así como las tecnologías, herramientas y entorno.

Capítulo II. Diseño de los elementos asociados al desarrollo de la solución propuesta: En este capítulo se realiza una descripción de la aplicación propuesta como solución y sus principales funcionalidades. Se lleva a cabo la disciplina de requisitos, donde son identificados los requisitos funcionales y no funcionales. Se argumenta el uso de los patrones de diseño, arquitectura, estándares de codificación y se detalla las particularidades de la metodología seleccionada.

Capítulo III. Validación de la solución propuesta: En este capítulo se comprueba que la propuesta de solución satisface el problema planteado y que se cumplen los objetivos de la investigación. Se lleva a cabo un proceso de validación de requisitos a través de las técnicas: revisión de los requisitos, diseño de prototipos y generación de casos de prueba. Se hace un análisis de los resultados obtenidos, se resuelven las no conformidades detectadas y se valora la viabilidad de la propuesta de solución.

CAPÍTULO I: REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO

En este capítulo se abordan los principales conceptos asociados a la investigación. Se realiza un estudio de las soluciones informáticas existentes para plataformas móviles con sistema operativo Android que permitirán conocer cómo se maneja la información en las mismas. Además, se realiza una comparación entre dichas soluciones para identificar las deficiencias de las mismas y lo que se puede tomar de ellas. Por último, se dan a conocer las tecnologías y la metodología para el desarrollo de la solución propuesta en el objetivo general.

1.1 Conceptos asociados al problema

Función notarial: es una función jurídica, legal y social, que cubre las necesidades existenciales del ser humano, con normas que tienden a la perfección de la vida en armonía, apunta al engranaje entre la vida de las personas y los bienes, determinando su posición jurídica respecto a ellos y hacia los terceros (Benítez, 2016).

Notarías: las funciones de una notaría son principalmente la de comprobar, legitimar y dar fe pública de los pactos concluidos entre personas. Todo esto por supuesto mediante la figura del Notario, que es la persona natural designada por la ley para cumplir las funciones mencionadas anteriormente (Benítez, 2016).

Servicios notariales: el servicio notarial es el conjunto de funciones que realiza un notario público (Araneda, 2015).

Dispositivos móviles: es un tipo de computadora de tamaño pequeño, con capacidades de procesamiento, con conexión a Internet, con memoria, diseñado específicamente para una función, pero que pueden llevar a cabo otras funciones más generales (Jiménez, 2023).

Aplicación móvil: es una aplicación desarrollada especialmente para ser ejecutada en dispositivos móviles como un teléfono celular o tablet (García & Mesa, 2019).

1.2 Análisis de soluciones similares

Después de haber realizado una investigación de los principales referentes teóricos relacionados con la problemática a resolver, se realiza un estudio de aplicaciones que mantienen relación con el contenido de la investigación, con el objetivo de definir características y viabilidad en el contexto nacional e internacional. Durante el estudio no se

encontraron aplicaciones internacionales similares que permitan personalizarse para facilitar el acceso a información notarial y solo se realizará un análisis a nivel Nacional.

iLex Asesoramiento Jurídico: es una herramienta móvil que constituye una propuesta del MINJUS para potenciar la informatización de la sociedad cubana. La plataforma está disponible para los sistemas operativos iOS y Android. Dentro de sus funcionalidades se incluye un extenso y actualizado directorio de instituciones. Entre ellas se cuentan departamentos provinciales, consultorías jurídicas y bufetes especializados de toda la Isla (Díaz, 2021).

iLex MINJUS: aplicación móvil desarrollada para facilitar la interacción entre el Ministerio de Justicia de la República de Cuba (MINJUS) y la población; permite a los ciudadanos el acceso a información de interés sobre la misión, funciones, estructura del organismo y los diferentes servicios que este brinda en el país; solicitar los certificados de antecedentes penales y de última voluntad y declaratoria de herederos; enviar quejas, inquietudes y planteamientos para ser tratados por la entidad (Pentón et al., 2021).

iLex Constitución: aplicación móvil desarrollada para facilitar la consulta de la Constitución de la República de Cuba, dado el desconocimiento de esta y la poca disponibilidad de ejemplares impresos (Pentón et al., 2021).

Callejero Jurídico: aplicación móvil desarrollada para brindar la ubicación geográfica de las entidades que ofrecen servicios jurídicos a la población: Bufetes colectivos, fiscalía general de la República (en todas sus instancias), Ministerio de Justicia de la República de Cuba, Sistema de Tribunales Populares de la República de Cuba (en todas sus instancias), Notarías, Registros y Consultorías Jurídicas. Además, el ciudadano puede consultar los servicios que brindan estas entidades, horarios, correos electrónicos y números telefónicos (Pentón et al., 2021).

iLex Notario: surgió por la necesidad de mostrar al público los servicios notariales de una forma más accesible como son los teléfonos móviles. Normalmente los ciudadanos tenían que dirigirse a las oficinas de las notarías para hacer preguntas y conocer sobre ciertos trámites lo que entorpecía el trabajo de los notarios y quitaba tiempo a los ciudadanos. Con iLex Notario los ciudadanos tienen desde la comodidad de su dispositivo móvil toda la información necesaria sobre los servicios notariales en el país (Pentón et al., 2021).

1.2.2 Conclusiones de las aplicaciones similares existentes

Tabla 1. Análisis de aplicaciones similares

Nombre aplicación	Accesibilidad	Información actualizada	Contiene información notarial
iLex Asesora-miento Jurídico	X	X	✓
iLex MINJUS	X	X	✓
iLex Constitución	X	X	X
Callejero Jurídico	X	X	✓
iLex Notario	X	X	✓

Fuente: Elaboración propia.

El objetivo de este trabajo es el desarrollo de una aplicación que permita mostrar al público los servicios notariales de una forma más accesible como son los teléfonos móviles. Las soluciones presentadas solo son empleadas como punto de partida para la investigación sobre todo las que están orientadas a brindar información notarial. De ellas se estudiaron los componentes más representativos de su interfaz y otras funcionalidades. Llegando a la conclusión que estas no cumplen con los indicadores necesarios para dar solución al problema planteado. En el caso de iLex Notario a pesar de cumplir con el indicador de información notarial, no presenta información actualizada provocando molestias en las entidades notariales correspondientes y confusiones en los usuarios a la hora de realizar los trámites que en la aplicación se mencionan, no permite modificar el tema de la interfaz de usuario y la fuente en cuanto tamaño y color, lo que dificulta la accesibilidad al contenido que ofrece la aplicación móvil a todo tipo de personas, aspectos que conllevan a la realización de una nueva versión de la aplicación. Se sugiere tomar como ejemplo las aplicaciones similares presentadas como base para la realización de la propuesta de solución, utilizando de estas las formas de organización, la estructura y las diferentes opciones que brindan. Por estas razones se decide desarrollar la versión 2.0 de la aplicación móvil iLex Notario para facilitar la consulta información notarial en el país y mejorar la accesibilidad a personas con distintas características y discapacidades.

1.3 Metodología de desarrollo

Las metodologías de ingeniería de software pueden considerarse como una base necesaria para la ejecución de cualquier proyecto de desarrollo de software que se considere serio, y que necesite sustentarse en algo más que la experiencia y capacidades de sus programadores y equipo. Estas metodologías son necesarias para poder realizar un proyecto profesional, tanto para poder desarrollar efectiva y eficientemente el software, como para que sirvan de documentación y se puedan rendir cuentas de los resultados obtenidos (Maida & Pacienzia, 2015).

Metodología de desarrollo de software AUP versión UCI

La Universidad de las Ciencias Informáticas (UCI) desarrolló una versión de la metodología de desarrollo de software AUP (Proceso Ágil Unificado), con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva de la universidad. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, nombrada Ejecución y agregándose también una nueva fase llamada Cierre (Alfonso, 2017).

La metodología de software AUP-UCI a partir del modelado de negocio propone tres variantes a utilizar en los proyectos, como son: CUN (Casos de uso del negocio), DPN (Descripción de proceso de negocio) o MC (Modelo conceptual) y existen tres formas de encapsular los requisitos las cuales son: CUS (Casos de uso del sistema), HU (Historias de usuario), DRP (Descripción de requisitos por proceso), surgen cuatro escenarios para modelar el sistema en los proyectos, los cuales son (Alfonso, 2017):

Escenario No 1: Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

Escenario No 2: Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

Escenario No 3: Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

De esta metodología se decide usar el escenario 4 debido a que este se aplica a proyectos que obtengan un negocio bien definido y no muy extensos, donde el cliente estará acompañando al equipo de desarrollo para convenir los detalles de los requisitos encontrados y poder implementarlos, probarlos y validarlos. Los requisitos se encapsulan en historias de usuarios.

1.4 Herramientas y tecnologías de desarrollo

1.4.1 Herramienta CASE

CASE (Ingeniería De Software Asistida Por Computador) esta incluye un conjunto de programas que facilitan la optimización de un producto ofreciendo apoyo permanente a los analistas, ingenieros de software y desarrolladores. Es la aplicación de métodos y técnicas que dan utilidades a los programas, por medio de otros, procedimientos y su respectiva documentación (Alarcón & Sandoval, 2008).

Visual Paradigm para UML v.8.0

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Agiliza la construcción de aplicaciones con calidad y a un menor coste (Hernández et al., 2016).

Se decide utilizar Visual Paradigm en su versión 8.0, teniendo en cuenta que esta herramienta propicia un conjunto de funcionalidades para el desarrollo de programas informáticos. Además, la UCI cuenta con la licencia para su utilización y soporta el ciclo completo del proceso de desarrollo de software

UML v.2.0

UML (Unified Modeling Language) es una notación de modelado visual que usa diagramas para mostrar distintos aspectos de un sistema. Es apto para modelar cualquier sistema, su mayor difusión y sus principales virtudes se advierten en el campo de los sistemas de software (Fontela, 2012).

Partiendo de que UML es la notación utilizada por la herramienta CASE, se decide la utilización de este como lenguaje de modelado.

1.4.2 Herramienta de prototipado

Pencil Project v.3.1.0

Pencil fue desarrollada con el objetivo de proporcionar una interfaz gráfica gratuita y de código abierto en la que se puedan realizar modelos y prototipos, y que se pueda instalar en múltiples plataformas. Pencil, proporciona varias colecciones de formas para crear diferentes tipos de interfaces, desde páginas web hasta aplicaciones móviles (Boisselier, 2021). Se utiliza la herramienta Pencil Project debido a que es una herramienta gratuita y de fácil manipulación.

1.4.3 Sistema operativo

A la hora de desarrollar una aplicación hay que prestarle atención al sistema operativo objetivo para el que se pretende desarrollar (Monterrubio, 2019):

- **Definición 1:** el sistema operativo es el principal programa que se ejecuta en toda computadora de propósito general. Los hay de todo tipo, desde muy simples hasta muy complejos, y entre más casos de uso hay para el cómputo en la vida diaria, más variedad habrá en ellos.
- **Definición 2:** un sistema operativo (SO) es un conjunto de rutinas o extensiones de software que permiten la administración y gestión del hardware. Consiste básicamente en rutinas de control que hacen funcionar una computadora y proporcionan un entorno para la ejecución de los programas.

A continuación, se muestra una gráfica comparativa de los sistemas operativos móviles más empleados en 2022 respectivamente.

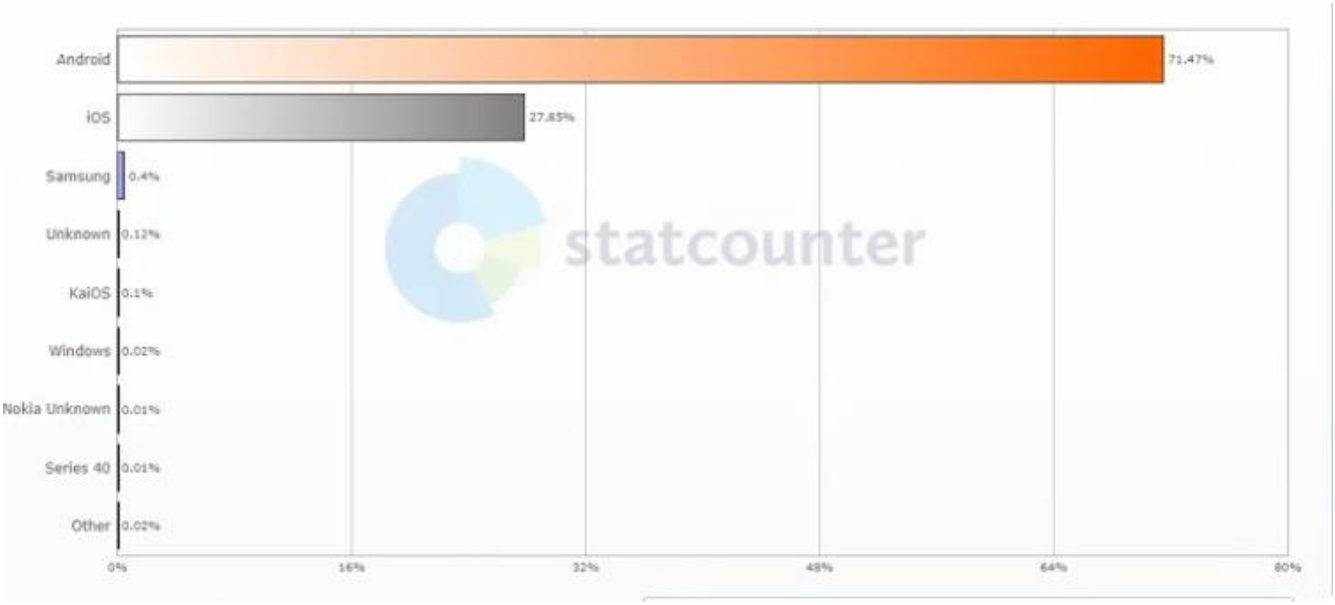


Figura 1. Comportamiento de android en el mercado

Fuente: (Ramírez, 2022)

Android mantuvo su posición como el sistema operativo líder a nivel mundial en 2022, controlando el mercado de sistemas operativos móviles con cerca del 72% de las acciones, mientras iOS se recuenta un 28% del mercado (Ramírez, 2022). Es por ello que se decide realizar este proyecto para dispositivos que utilicen Android porque la propuesta solución es una aplicación móvil para este sistema, con el objetivo de que llegue a la mayor cantidad de personas posibles, además de ser la segunda versión de una aplicación existente.

1.4.4 Entorno de Desarrollo Integrado (IDE)

Un IDE, o entorno de programación, se define como la aplicación que se utiliza para editar, compilar, depurar y construir la interfaz gráfica de nuestro código de una forma amigable. En un inicio estas aplicaciones se podrían utilizar solamente a nivel de escritorio, pero conforme ha evolucionado la forma de almacenar la información se ha comenzado a migrar dichos entornos a sus propias versiones basadas en navegadores web con las mismas características que los basados en escritorio (Patterson, 2015).

Visual Studio Code v. 1.78.2

Visual Studio Code (VS Code) es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones

para otros lenguajes y tiempos de ejecución (como C ++, C #, Java, Python, PHP, Go, .NET).Entre sus ventajas se pueden enumerar (Visual Studio Code, 2023):

- Permite habilitar idiomas, temas, depuradores, comandos y más funcionalidades para mejorar el flujo de trabajo.
- Proporciona VS Code IntelliSense para JavaScript, TypeScript, JSON, HTML, CSS, SCSS y Less. Admite completaciones basadas en palabras para cualquier lenguaje de programación, pero también se puede configurar para tener un IntelliSense más completo instalando una extensión de lenguaje.
- Es un excelente soporte de depuración. El depurador integrado de VS Code ayuda a acelerar su ciclo de edición, compilación y depuración.
- VS Code contiene las funciones necesarias para una edición de código fuente altamente productiva.

1.4.5 Lenguaje de Programación

Desde un punto de vista coloquial, un lenguaje de programación es una notación para comunicarle a una computadora lo que deseamos que haga. Desde un punto formal, se puede definir como un sistema notacional para describir computaciones en una forma legible tanto para la máquina como para el ser humano (Bellas et al., 2016).

Dart 3.6.4

Dart es un lenguaje optimizado para el cliente para desarrollar aplicaciones rápidas en cualquier plataforma. Su objetivo es ofrecer el lenguaje de programación más productivo para desarrollo multiplataforma, junto con una plataforma de tiempo de ejecución flexible para marcos de aplicaciones. Está diseñado para un sobre técnico que es especialmente adecuado para el desarrollo de clientes, priorizar tanto el desarrollo (recarga en caliente con estado inferior a un segundo) como experiencias de producción de alta calidad en una amplia variedad de destinos de compilación (web, móvil y de escritorio). También forma la base de Flutter, proporciona el lenguaje y los tiempos de ejecución que impulsan las aplicaciones de Flutter, pero Dart también admite muchas tareas básicas de desarrollo como dar formato, analizar y probar código (Dart, 2023a). Se utiliza este lenguaje debido a que Dart es el lenguaje de programación oficial de Flutter.

1.4.6 Framework

Un *framework* se compone de un conjunto de clases y de flujos de control y de esta manera, provee una estructura precisa para definir nuevas aplicaciones que resuelven problemas dentro de un dominio dado (Roa et al., 2008).

Flutter 3.10.2

Flutter es el kit de herramientas de UI de Google para realizar aplicaciones, compiladas nativamente, para móvil, web y escritorio desde una única base de código (Flutter, 2023). Entre sus ventajas se pueden enumerar:

Desarrollo Rápido: El *hot reload* de Flutter te ayuda a rápida y fácilmente experimentar, construir UIs, añadir funcionalidades, y corregir bugs más rápido. Experimenta tiempos de recarga por debajo de un segundo, sin perder el estado, en emuladores, simuladores, y dispositivos para iOS y Android (Flutter, 2023).

UI Expresiva y Flexible: Ejecuta rápidamente funcionalidades con el foco en la experiencia de usuario nativa. La arquitectura en capas permite una completa personalización, que resultan en un renderizado increíblemente rápido y diseños expresivos y flexibles (Flutter, 2023).

Rendimiento Nativo: Los widgets de Flutter incorporan todas las diferencias críticas entre plataformas, como el *scrolling,* navegación, iconos y fuentes para proporcionar un rendimiento totalmente nativo tanto en iOS como en Android (Flutter, 2023).

Por estas razones se decide utilizar Flutter para el desarrollo de la propuesta solución, además de ser el definido por el equipo de desarrollo del proyecto LexCuba.

1.4.7 Kit de desarrollo de software (SDK)

“Un kit de desarrollo de software (SDK por sus siglas en inglés) es un conjunto de herramientas proporcionado usualmente por el fabricante de una plataforma de hardware, un sistema operativo (SO) o un lenguaje de programación. Los SDK permiten que los desarrolladores de software creen aplicaciones para esa plataforma, ese sistema o ese lenguaje de programación específicos (González, 2021).

SDK de Android

Son múltiples paquetes que permiten al programador escribir código, recuperar componentes de las pantallas, hacer llamadas a paquetes, detectar errores etc. Todo de forma automática haciendo que la programación sea más fácil (González, 2021).

1.4.8 Control de versiones

Los sistemas de control de versiones se encargan de almacenar el historial de las modificaciones que va sufriendo los diferentes archivos de un proyecto, es decir, de gestionar los sucesivos cambios que se realizan sobre los elementos de algún producto o una configuración de mantenimiento de manera estructurada de los avances, retrocesos y modificaciones del software mientras está siendo desarrollado (Abrutsky, 2016).

GitLab v.16.3.3

GitLab reúne todas las capacidades de DevSecOps en una aplicación con un almacén de datos unificado para que todo esté en un solo lugar. Ofrece una experiencia de usuario superior, lo que mejora el tiempo de ciclo y ayuda a evitar el cambio de contexto. Las herramientas de automatización de GitLab son más confiables y ricas en funciones, lo que ayuda a eliminar la carga cognitiva y el trabajo pesado innecesario (GitLab, 2023).

1.4.9 JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. JSON está construido por dos estructuras (JSON, 2023):

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra, por estas razones son utilizados como ficheros para almacenar las normas en la implementación de la aplicación.

Conclusiones del capítulo

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- El análisis de los conceptos fundamentales asociados al proceso de consulta de información notarial permitió adquirir una mejor comprensión de los términos asociados al problema.
- A través del estudio de los sistemas homólogos se comprobó que ninguno de esos sistemas soluciona el problema, por lo que esto será resuelto con la propuesta de solución.
- La metodología de desarrollo del software AUP variación UCI seleccionada, hará del producto de salida un software de mayor calidad satisfaciendo las necesidades del cliente.
- El estudio realizado al ambiente de desarrollo permitió seleccionar las herramientas y tecnologías adecuadas a utilizar para la implementación de la propuesta de solución, logrando así el cumplimiento de los objetivos planteados.

CAPÍTULO II: DISEÑO DE LOS ELEMENTOS ASOCIADOS AL DESARROLLO DE LA PROPUESTA SOLUCIÓN

En el presente capítulo se realiza la descripción y diseño de la propuesta de solución. Además, se lleva a cabo el proceso de levantamiento de información, donde se identifican los requisitos funcionales y no funcionales, se generan las historias de usuario a partir de los requisitos funcionales. Se describen la arquitectura, los patrones de diseño y los estándares de codificación utilizados.

2.1 Descripción de la propuesta de solución

En el presente trabajo se propone desarrollar iLex Notario 2.0 con el objetivo de mejorar la consulta de información sobre las notarías en Cuba. La aplicación contará con varias secciones: la sección notas introductorias, permitirá consultar las finalidades que se cumplen con la elaboración de la aplicación; por otra parte, la sección de información de marco legal, permitirá consultar información referente al marco legal notarial; la sección de unidades notariales, brindará acceso a la información de las unidades notariales perteneciente a las sociedades civiles de derecho jurídico; y la sección de directorios, mostrara información de contacto así como de ubicación de las notarías por provincias. De igual manera, presentará un espacio de configuración, donde se puede cambiar el tema, tamaño y color de letra, además de un menú lateral que ofrece acceso a otras funcionalidades: como información acerca de la aplicación, de la institución (MINJUS) y algunos enlaces de interés.

2.2 Requisitos de Software

Los requisitos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requisitos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito (Sommerville, 2011).

Luego de un estudio realizado de las técnicas existentes para obtener/recopilar requisitos de software, se decide utilizar la técnica de las entrevistas (Anexo 1). Las entrevistas pueden ser abiertas o cerradas. No están consideradas como una técnica muy eficaz, pero son útiles para complementar otras técnicas de obtención de requisitos (Estévez, 2020).

2.2.1 Requisitos funcionales del sistema

Son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas (Sommerville, 2011). A continuación, en la tabla se muestran los 18 requisitos funcionales del sistema identificados, siendo 3 de estos requisitos de complejidad alta, 13 requisitos de complejidad media y 2 de complejidad baja, 17 de prioridad alta, 1 de prioridad media.

Tabla 2. Especificación de requisitos funcionales

No.	Nombre	Descripción	Complejidad	Prioridad
RF1	Mostrar nota Introductoria	La aplicación permitirá mostrar información acerca de las finalidades que se cumplen con el desarrollo de la aplicación y de la actuación notarial.	Media	Alta
RF2	Mostrar información del notario	La aplicación permitirá mostrar información sobre las funciones y servicios que realizan los notarios en Cuba.	Media	Alta
RF3	Mostrar información de la función notarial en la República de Cuba	La aplicación permitirá mostrar información sobre la función notarial en la República de Cuba y de cómo se comprende la legislación notarial.	Media	Alta
RF4	Listar marco legal notarial	La aplicación permitirá mostrar la lista de	Alta	Alta

		normas asociadas a la actividad notarial en Cuba.		
RF5	Mostrar norma	La aplicación permitirá mostrar información acerca de la norma seleccionada.	Media	Alta
RF6	Listar provincias con unidades notariales	La aplicación permitirá mostrar la lista de provincias y sus unidades notariales.	Alta	Alta
RF7	Mostrar unidad notarial	La aplicación permitirá mostrar información de contacto y geográfica sobre la unidad seleccionada.	Media	Alta
RF8	Listar unidades notariales pertenecientes a las sociedades civiles de derecho jurídico	La aplicación permitirá mostrar la lista de unidades notariales pertenecientes a las sociedades civiles de derecho jurídico.	Alta	Alta
RF9	Mostrar unidad notarial pertenecientes a las sociedades civiles de derecho jurídico	La aplicación permitirá mostrar información de contacto y geográfica sobre la unidad seleccionada.	Media	Alta
RF10	Mostrar menú lateral	La aplicación permitirá desplegar un menú lateral con enlaces de	Media	Alta

		interés del MINJUS y otras funcionalidades.		
RF11	Mostrar servicios notariales	La aplicación permitirá mostrar información sobre los servicios notariales que ofrecen las notarías del país.	Media	Alta
RF12	Mostrar documentos que se redactan	La aplicación permitirá mostrar información sobre los documentos que se redactan en las notarías.	Media	Alta
RF13	Mostrar deberes y derechos	La aplicación permitirá mostrar información sobre deberes del notario y derechos de los ciudadanos.	Media	Alta
RF14	Mostrar dirección de notaría	La aplicación permitirá mostrar información de contacto de la dirección de notarias del MINJUS.	Media	Alta
RF15	Mostrar correos	La aplicación permitirá mostrar información de contacto de correo electrónico de los directores de las unidades notariales.	Media	Alta
RF16	Modificar fuente	La aplicación permitirá cambiar el tamaño y color de la fuente.	Baja	Alta

RF17	Modificar tema	La aplicación permitirá cambiar el tema de la aplicación.	Baja	Alta
RF18	Mostrar información de la aplicación	La aplicación permitirá mostrar información de contacto y soporte de la aplicación.	Media	Media

Fuente: Elaboración propia.

2.2.2 Requisitos no funcionales del sistema

Los requerimientos no funcionales, como indica su nombre, son requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios. Pueden relacionarse con propiedades emergentes del sistema, como fiabilidad, tiempo de respuesta y uso de almacenamiento (Sommerville, 2011). A continuación, se enumeran los requisitos no funcionales con sus clasificaciones pertenecientes a la propuesta de solución:

1. **Rendimiento**

   RnF 1: El sistema debe ejecutarse de forma correcta en dispositivos con al menos 1 gigabyte de memoria RAM.

2. **Usabilidad**

   RnF 2: El sistema debe ejecutarse en sistemas con una versión del Sistema Operativo Android mayor de 4.1 JellyBean.

   RnF 3: El sistema debe usar los colores de la entidad representada.

   RnF 4: Las interfaces deben poseer un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad.

   RnF 5: En el sistema se deben visualizar todos los mensajes en idioma español.

   RnF 6: Se debe poder alternar entre tema oscuro y tema claro.

3. **Desarrollo**

   RnF 7: El sistema será desarrollado en el lenguaje de programación Dart usando el *framework* Flutter.

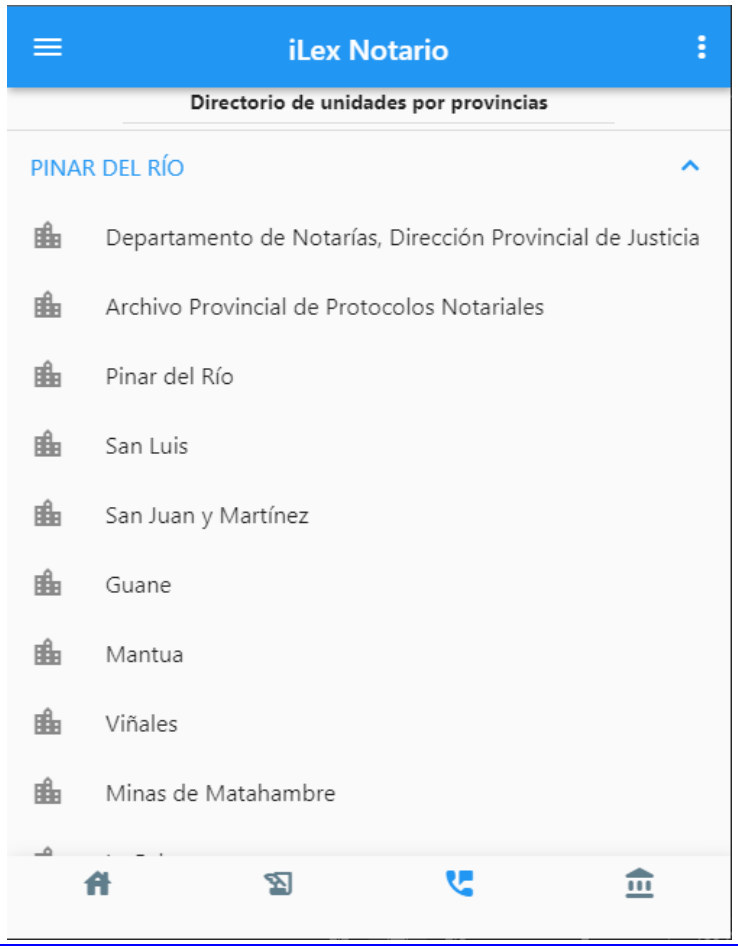
2.3 Historias de Usuario

Las historias de usuario se usan, en el contexto de la ingeniería de requisitos ágil, como una herramienta de comunicación que combina las fortalezas de ambos medios: escrito y verbal. Describen, en una o dos frases, una funcionalidad de software desde el punto de vista del usuario, con el lenguaje que éste emplearía. El foco está puesto en qué necesidades o problemas soluciona lo que se va a construir (Menzinsky et al., 2022). En total se definieron 18 HU, a continuación, se describe la perteneciente al RF 6 debido a su prioridad para el cliente. El resto de las historias de usuarios podrán ser consultadas en los documentos adjuntos.

Tabla 3. HU Listar provincias con unidades notariales

Número: 6	**Requisito**: Listar provincias con unidades notariales
Programador: Frey Fernando Griñan Despaigne	**Iteración Asignada**: 1
Prioridad: Alta	**Tiempo Estimado**: 10 días
Riesgo en Desarrollo: N/A	**Tiempo Real**: 7 días
Descripción: funcionalidad que permitirá visualizar la lista de provincias con sus unidades notariales. Botón Menú Lateral: permite desplegar un menú lateral. Botón de menú emergente: permite visualizar las opciones de configuración e información de la aplicación. Barra de navegación: permite navegar por las secciones de la aplicación.	
Observaciones:	

Prototipo elemental de interfaz gráfica de usuario:



Fuente: Elaboración propia.

2.4 Arquitectura del Sistema

La Arquitectura de Software se ha consolidado como una disciplina que intenta contrarrestar los efectos negativos que pueden surgir durante el desarrollo de un software, ocupando un rol significativo en la estrategia de negocio que basa sus operaciones en el software (Rodríguez Peña & Silva Rojas, 2016). La arquitectura definida para la propuesta solución es:

La Arquitectura Limpia (Clean Architecture) es una opción sólida para estructurar el proyecto. Con un enfoque en la separación de capas y la dependencia unidireccional, la que permitirá mantener un código más organizado y fácil de mantener. Además, ofrece la ventaja de mitigar el impacto de futuros cambios en la aplicación (Arias, 2022).

Está basada en 5 principios fundamentales (Martínez et al., 2021):

- Independiente de cualquier *framework*: debe ser aplicable en cualquier sistema sin importar el lenguaje de programación o las librerías que se utilice. Las capas deben estar bien separadas, de modo que puedan sobrevivir individualmente sin necesidad de externos.
- Testeable: cada módulo, tanto de UI, base de datos, conexión a API Rest, etc., se debe poder testear de forma individual.
- Independiente de la Interfaz de usuario (UI): la UI es capaz de cambiar sin alterar todo el sistema y debe vivir tan independiente que podría ser desensamblada y sustituida por otra. Por ejemplo, cambiar una UI Móvil por una en modo consola.
- Independiente de la base de datos: esta capa debe ser tan modular que sea posible agregarle múltiples fuentes de datos, e incluso múltiples fuentes del mismo tipo de datos. Por ejemplo, manejar varias bases de datos como MySQL, PostgreSQL, Redis, etc.
- Independiente de cualquier elemento externo: si en algún punto se necesitara una librería, otro sistema o cualquier elemento a conectar, debería ser modularizado. Para el sistema esta capa externa debería ser transparente.

En la figura 3 se evidencia el empleo de la arquitectura Clean Architecture en el desarrollo de la propuesta solución. Separando responsabilidades en distintas capas y relacionando las capas aledañas. Donde la capa de presentación interactúa con la interfaz de usuario mediante el patrón de diseño MVP (Modelo-Vista-Presentador). La segunda capa es la del dominio que se encarga de contener la lógica de negocio y de almacenar mediante los casos de uso. Por último, la capa de datos es donde están los datos reservados en JSON y donde se acceden a los mismos y se implementan los repositorios para trabajar con los modelos.

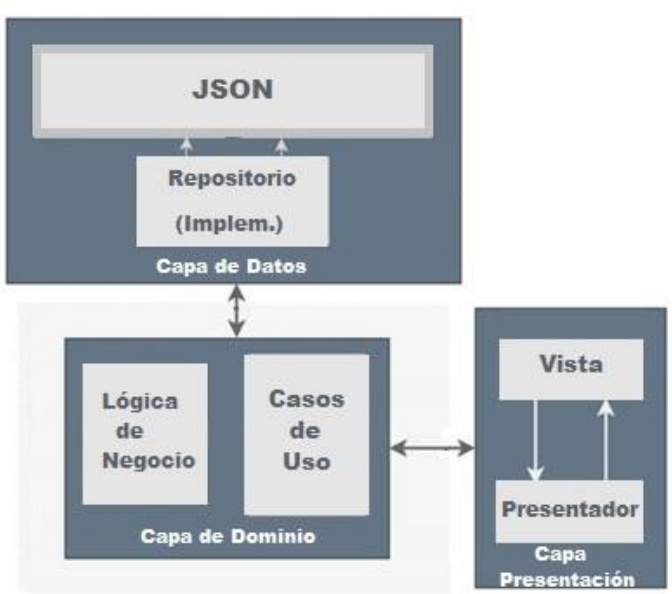


Figura 2. Estructura de la implementación de la arquitectura limpia

Fuente: Elaboración propia.

2.4.1 Diagrama de la arquitectura

A continuación, se muestra el diagrama de la arquitectura empleado en el desarrollo de la aplicación móvil: la capa de presentación implementa los elementos visuales a través de widgets y mediante la lógica de la presentación realiza el llamado de los casos de uso. La capa de dominio es donde se encapsulan e implementan todos los casos de uso del sistema y la abstracción del repositorio. La capa de datos es donde se implementa el repositorio y los modelos los cuales contienen el código para procesar los datos almacenados en la fuente de datos local.

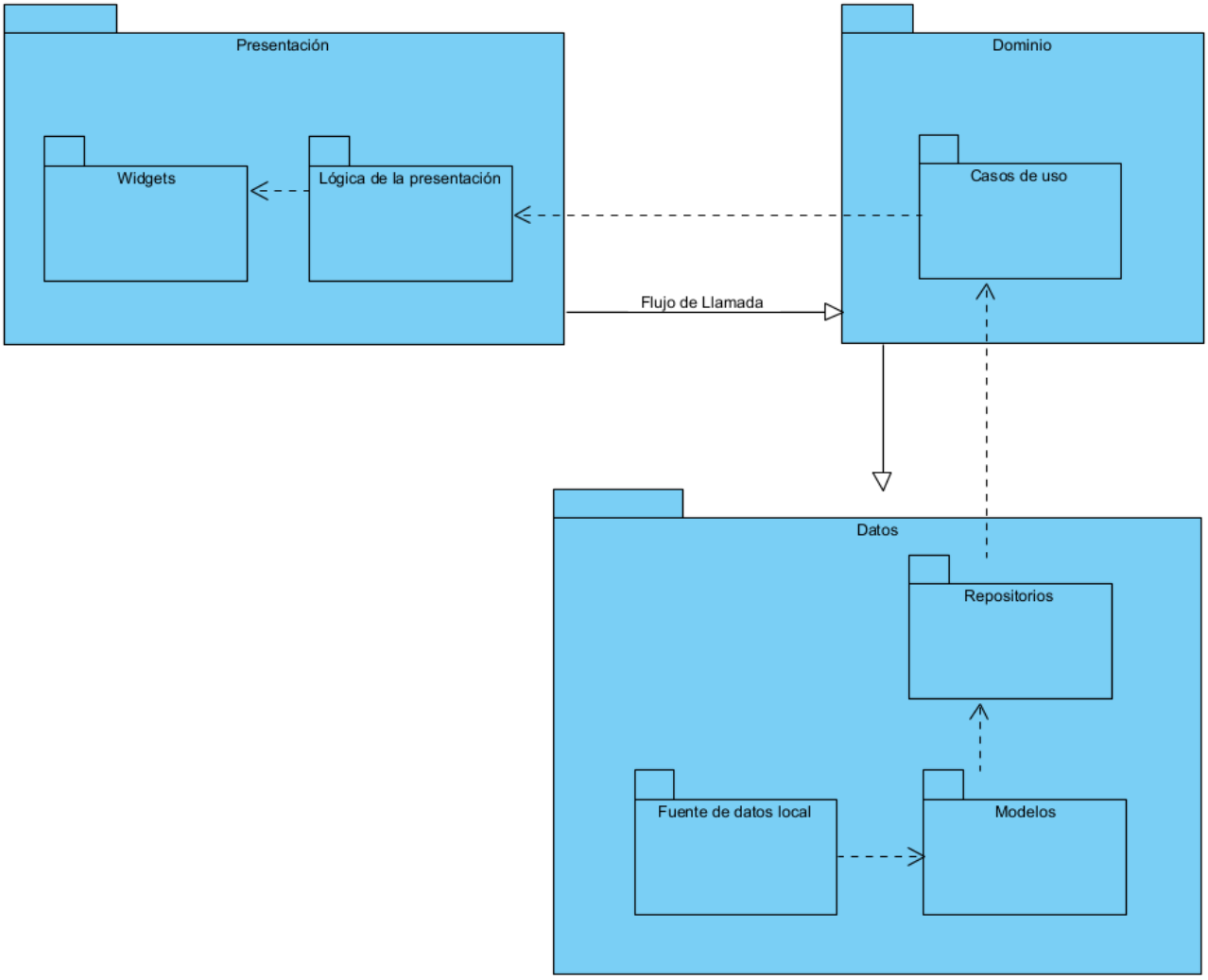


Figura 3. Diagrama de la arquitectura

Fuente: Elaboración propia.

2.4.2 Patrones de diseño

Los patrones de diseño son técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Pressman, 2010).

2.4.1.1 Patrones GRASP

Los Patrones de Principios Generales para Asignar Responsabilidades (GRASP), más que patrones propiamente dichos, son una serie de buenas prácticas de aplicación recomendable en el diseño de software (Tabares, 2010). A continuación, se describen los utilizados en la propuesta de solución:

El patrón **Creador** aporta un principio general para la creación de objetos, una de las actividades más frecuentes en programación (Tabares, 2010). Este patrón puede evidenciarse en la creación de clases de tipo “Entidad” y “Widget”:

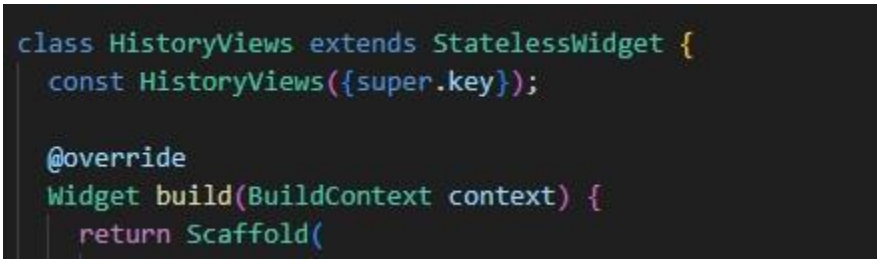
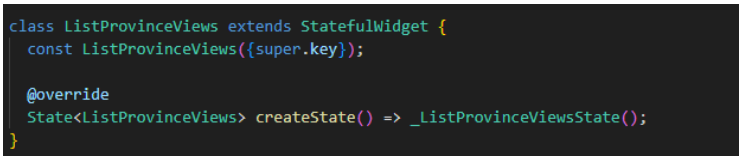


Figura 4. Clase "HistoryViews" evidenciando el patrón “Creador”

El patrón **Bajo Acoplamiento** es una medida de la fuerza con que una clase se relaciona con otras, porque las conoce y recurre a ellas; una clase con bajo acoplamiento no depende de muchas otras, mientras que otra con alto acoplamiento presenta varios inconvenientes: es difícil entender cuando está aislada, es ardua de reutilizar porque requiere la presencia de otras clases con las que esté conectada y es cambiante a nivel local cuando se modifican las clases afines (Tabares, 2010).

El patrón **Alta Cohesión** es una medida que determina cuán relacionadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo colosal; una clase con baja cohesión realiza un trabajo excesivo, haciéndola difícil de comprender, reutilizar y conservar (Tabares, 2010).

A continuación, se evidencian el uso de los patrones bajo acoplamiento y alta cohesión donde la clase “ProvinceUnitInfo” implementa los métodos de la clase “ListProvinceViews”.



```dart
Widget _listprovince() {
  return FutureBuilder<List<ArrayProvincia>>(
    future: provinceProvider.getprovince(),
    initialData: const [],
    builder: (context, AsyncSnapshot<List<ArrayProvincia>> snapshot) {
      return Center(
        child: ListView.builder(
          itemBuilder: (context, index) {
            return ExpansionTile(
              title: Text(snapshot.data![index].nombreProvincia ?? '--'),
              children: [
                ListView.builder(
                  shrinkWrap: true,
                  physics: const ScrollPhysics(),
                  itemCount: snapshot.data![index].arrayUnidades.length,
                  itemBuilder: (context, indexSon) {
                    return ListTile(
                      onTap: () {
                        Get.to(ProvinceUnitInfo(
                          arrayunidades:
                              snapshot.data![index].arrayUnidades[indexSon],
                        )); // ProvinceUnitInfo
                      },
                      leading: const Icon(Icons.location_city),
                      title: Text(
                        snapshot.data![index].arrayUnidades[indexSon]
                                .unidad ??
                            '--',
                      ), // Text
                    ); // ListTile
```

```dart
class ProvinceUnitInfo extends StatelessWidget {
  const ProvinceUnitInfo({super.key, required this.arrayunidades});

  final ArrayUnidad arrayunidades;
```

```dart
                Card(
                  elevation: 10,
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                    const Text("Unidad:"),
                    Text(arrayunidades.unidad ??  '  '),
                    const SizedBox(height: 8.0),
                    const Text("Direccion:"),
                    Text(arrayunidades.direccion ??  '  '),
                      ],
                    ), // Column
                  ), // Card
```

Figura 5. Clase" ListProvinceViews " y Clase " ProvinceUnitInfo " evidenciando el patrón “Bajo acoplamiento” y de “Alta cohesión”.

2.4.1.2 Patrones Gof

El proceso de identificación de patrones de diseño GOF en procesos de desarrollo software requiere de un conjunto de actividades; inicialmente se construye un conjunto de criterios que permita establecer un punto de análisis para identificar los procesos de desarrollo más representativos, siendo obligatorio cumplir estos criterios para catalogarlos como válidos en este análisis, y obtener la muestra de procesos de desarrollo (Guerrero et al., 2013). A continuación, se describen los utilizados en la propuesta de solución:

Patrón Observador: es uno de los más conocidos y al implementarlo en forma de componente se obtienen beneficios importantes. Es necesario tener clara la idea de lo que significa un componente: una unidad de composición con interfaces especificadas contractualmente. Estas interfaces establecen condiciones entre el proveedor del servicio y el usuario y son el punto de acceso de los usuarios para acceder al servicio que se presta (Guerrero et al., 2013).

En las siguientes imágenes se muestra una clase proveedora de la que va a depender la capa de presentación y la implementación de la proveedora, la capa de presentación está separada de la lógica de las proveedoras.

```
class ProvinceProvider {
  ProvinceProvider();

  Future<List<ArrayProvincia>> getprovince() async {
    final resp = await rootBundle.loadString('data/directorio_notario.json');
    return DirectoryProvince.fromJson(json.decode(resp)).arrayProvincias;
  }
}

final provinceProvider = ProvinceProvider();
```

```
class ListProvinceViews extends StatefulWidget {
  const ListProvinceViews({super.key});

  @override
  State<ListProvinceViews> createState() => _ListProvinceViewsState();
}
```

```
Widget _listprovince() {
  return FutureBuilder<List<ArrayProvincia>>(
    future: provinceProvider.getprovince(),
    initialData: const [],
    builder: (context, AsyncSnapshot<List<ArrayProvincia>> snapshot) {
      return Center(
        child: ListView.builder(
          itemBuilder: (context, index) {
            return ExpansionTile(
              title: Text(snapshot.data![index].nombreProvincia ?? '--'),
              children: [
                ListView.builder(
                  shrinkWrap: true,
                  physics: const ScrollPhysics(),
                  itemCount: snapshot.data![index].arrayUnidades.length,
                  itemBuilder: (context, indexSon) {
                    return ListTile(
                      onTap: () {
                        Get.to(ProvinceUnitInfo(
                          arrayunidades:
                              snapshot.data![index].arrayUnidades[indexSon],
                        )); // ProvinceUnitInfo
                      },
                      leading: const Icon(Icons.location_city),
                      title: Text(
                        snapshot.data![index].arrayUnidades[indexSon]
                            .unidad ??
                        '--',
                      ), // Text
                    ); // ListTile
```

Figura 6. Implementación que permite el patrón "Observador"

2.5 Estándares de Codificación

El uso de estándares de codificación, es una práctica altamente recomendada para desarrollar software de alta calidad, los estándares permiten establecer criterios únicos que los programadores deben implementar cuando escriben código, al utilizar un estándar de codificación se busca que el código fuente pueda ser entendido por cualquier miembro del equipo de desarrollo, esto permite que el código pueda ser modificado por otro programador evitando así la dependencia de personal o en el peor de los casos tener que volver a escribir la totalidad del código, lo que ocasionaría costos extras y mayor tiempo del requerido (Castellanos Rodríguez & Osorio Franco, 2018). Para el desarrollo de la investigación se siguieron los estándares presentes en el sitio oficial del lenguaje Dart entre los cuales se puede mencionar (Dart, 2023b):

1. Identificadores:

- UpperCamelCase: los nombres se escriben en mayúscula, la primera letra de cada palabra, incluyendo la primera.
- lowerCamelCase: los nombres se escriben en mayúscula, la primera letra de cada palabra, excepto la primera que siempre está en minúsculas, incluso si se trata de un acrónimo.
- lowercase_with_underscores: los nombres usan solo letras minúsculas, incluso para los acrónimos, y separar las palabras con _.

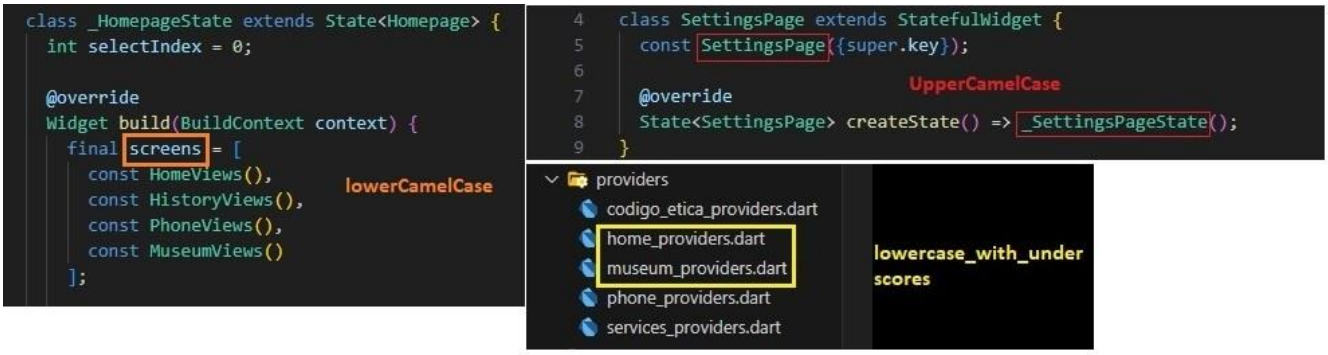


Figura 7. Ejemplo de identificadores

2. Orden:

• Colocar las exportaciones en una sección separada después de todas las importaciones.

• Colocar las importaciones antes que otras importaciones “dart:”.

• Colocar las importaciones antes que las importaciones relativas “package:”.

• Ordenar las secciones alfabéticamente.

```dart
import 'dart:convert';

import 'package:flutter/services.dart' show rootBundle;
import 'package:ilex_notario_app/src/models/directorio_notario_model.dart';
```

```dart
import 'package:flutter/material.dart';

import 'package:ilex_notario_app/src/pages/views/historyviews.dart';
import 'package:ilex_notario_app/src/pages/views/homeviews.dart';
import 'package:ilex_notario_app/src/pages/views/museumviews.dart';
import 'package:ilex_notario_app/src/pages/views/phoneviews.dart';
```

Figura 8. Ejemplo de Importaciones

3. Tamaño sugerido de líneas:

• Evitar las líneas de código de más de 80 caracteres.

```dart
const DirectoryInfo({super.key, required this.arrayunidades});
```

Figura 9. *Ejemplo de tamaño sugerido de líneas, con 78 caracteres*

4. Uso de llaves:

• Utilice llaves para todas las instrucciones de control de flujo.

```dart
onSelected: (value) {
  if (value == 2) {
    Navigator.of(context).pushNamed('settings');
  } else if (value == 1) {
    //navegar al acerca de
  }
},
```

Figura 10. Ejemplo de uso de llaves

Conclusiones del capítulo

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- El empleo de la metodología AUP en su variación para la UCI, permitió generar los productos de trabajo necesarios pertenecientes a la disciplina de análisis y diseño, así como una mayor organización en el desarrollo de la propuesta.

- El levantamiento de información permitió obtener los requisitos funcionales y los requisitos no funcionales, así como la realización de las historias de usuarios a partir de los requisitos funcionales.
- La arquitectura del sistema Clean Architecture permitió una mayor organización de la solución, así como la ampliación de las funcionalidades de esta.
- Los patrones de diseño y estándares de codificación definidos permitieron sentar las normas a seguir durante todo el ciclo de la implementación, garantizando la obtención de una solución con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios.

CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En este capítulo se evalúa el grado de fiabilidad y calidad de los requisitos obtenidos. Además, se aplican pruebas de software y métodos que evalúan el código fuente y las posibles entradas y salidas del sistema para verificar y revelar la calidad de la solución propuesta antes de ser entregada al cliente.

3.1 Técnicas de Validación de requisitos

La validación de requisitos es el proceso de verificar que los requisitos definan realmente el sistema que en verdad quiere el cliente. Se cubre con el análisis, ya que se interesa por encontrar problemas con los requisitos. (Sommerville, 2011). Con el objetivo de validar los requisitos se emplearon técnicas que se mencionan a continuación:

- Revisiones de requisitos: los requisitos se analizan sistemáticamente usando un equipo de revisiones que verifican errores e inconsistencias (Sommerville, 2011).
- Creación de prototipos: en esta aproximación a la validación, se muestra un modelo ejecutable del sistema en cuestión a los usuarios finales y clientes. Así, ellos podrán experimentar con este modelo para constatar si cubre sus necesidades reales (Sommerville, 2011).

Luego de aplicar las técnicas mencionadas anteriormente se obtuvieron los resultados que a continuación se describen:

3.1.1 Creación de prototipos

Se mostró al cliente un modelo ejecutable que permitió tener una visión preliminar de cómo se verían estos componentes en el sistema y a través de la interacción con los mismos se comprobó la satisfacción y aprobación del cliente, los mismos fueron aprobados satisfactoriamente.

3.1.2 Aplicación de la métrica Calidad de especificación

Con el objetivo de medir la calidad de la especificación de los requisitos se aplicó una de las métricas propuestas para la metodología AUP (UCI), Calidad de la especificación (CE). A fin de obtener cuán entendibles y precisos son los requisitos, para ello se calcula el total de requisitos de la especificación como se muestra a continuación:

Nr: Total de requisitos de software

Nf: Cantidad de requisitos funcionales

Nnf: Cantidad de requisitos no funcionales

Nr = Nf + Nnf

Nr = 18 + 7

Nr = 25

Finalmente, para determinar la Especificidad de los Requisitos (ER) se tuvo en cuenta la ausencia de ambigüedad en los mismos (mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad), para ello se realiza la siguiente operación:

ER = Nui / Nr

Nui: número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas.

Para sustituir los valores de las variables en la ecuación se tuvo en cuenta que, de los requisitos especificados para el desarrollo de la propuesta de solución, dos de ellos causaron contradicción en sus interpretaciones. Por tanto, la variable ER obtiene el siguiente valor:

ER = 23 / 25

ER = 0.92

Teniendo en cuenta el valor anterior, se arroja a un resultado positivo donde el grado de ambigüedad de los requisitos es bajo (8%) ya que el 92% son entendibles. Los requisitos ambiguos fueron modificados y validados para garantizar una correcta interpretación, llegando al resultado ideal de ER=1.

3.2 Pruebas de software

Proceso que consiste en todas las actividades del ciclo de vida software, tanto estáticas como dinámicas, concernientes con la planificación, preparación y evaluación de productos software y los productos de trabajo relacionados para determinar que éstos satisfacen los requisitos especificados, para demostrar que se ajustan al propósito y para detectar defectos (Sánchez, 2015).

Este proceso de prueba conduce a dos metas diferentes:

- Demostrar al desarrollador y al cliente que el software cumple con los requisitos.
- Encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación (Sánchez, 2015).

3.2.1 Pruebas a nivel de unidad

Las pruebas unitarias se realizan sobre las funcionalidades internas de un módulo y se encargan de comprobar los caminos lógicos, ciclos y condiciones que debe cumplir el programa (Sommerville, 2011).

Para aplicar las pruebas unitarias se decide utilizar el método de caja blanca, ya que posibilita desarrollar casos de prueba que garanticen la ejecución, al menos una vez, de las rutas independientes (Sánchez, 2015). Para aplicar este método se define la técnica de ruta básica.

Técnica de ruta básica: tiene como objetivo comprobar que cada camino se ejecute independientemente de un componente o programa, obteniéndose una medida de la complejidad lógica del diseño. Esta técnica de prueba debe ser utilizada para evaluar la efectividad de los métodos asociados a una clase, con el objetivo de asegurar que cada ruta independiente sea ejecutada por lo menos una vez en el sistema (Pressman, 2010).

Se propone realizar un análisis de la complejidad ciclomática de cada procedimiento que componen las clases del sistema; una vez concluido este paso se selecciona el método con valor de contener errores, además de que ofrece una medida del número de pruebas que deben diseñarse para validar la correcta implementación de una determinada función, según la estrategia de Pressman para aplicar la ruta básica.

Para obtener la cantidad de mínima de casos de prueba a realizar a partir de la técnica ruta básica, se debe construir el grafo de flujo para cada uno de los procedimientos desarrollados, a continuación, se muestra un ejemplo de la aplicación de la técnica correspondiente al código del botón de acceso a los ajustes de la aplicación y a la información de la misma.

Figura 11. PopupMenuButton() utilizado como ejemplo para la técnica de ruta básica.

Fuente: Elaboración propia.

Una vez definido el código sobre el cual se aplica el método, los pasos a seguir para desarrollar la técnica de ruta básica son los siguientes:

1) Confeccionar el grafo de flujo: este muestra el flujo de control lógico(Pressman, 2010). Está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

En la siguiente ilustración, se muestra el grafo de flujo obtenido:

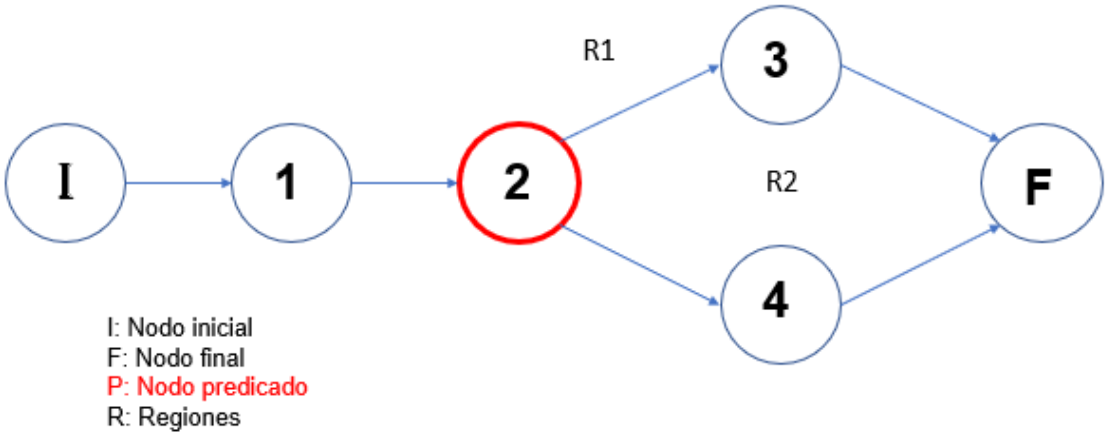


Figura 12. Grafo del flujo PopupMenuButton().

Fuente: Elaboración propia.

2) Calcular la complejidad ciclomática:

El valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa y brinda una cota superior para el número de pruebas que se deben realizar a fin de asegurar que todos los enunciados se ejecutaron al menos una vez (Pressman, 2010).

La complejidad ciclomática se calcula de tres formas diferentes, las cuales deben llegar al mismo resultado para comprobar que el cálculo es el correcto (Pressman, 2010).

- El número de regiones del grafo de flujo corresponde a la complejidad ciclomática.
- La complejidad ciclomática V(G) para un grafo de flujo G también se define como V(G)=E-N+2, donde E es el número de aristas del grafo de flujo y N el número de nodos del grafo de flujo.
- La complejidad ciclomática V(G) para un grafo de flujo G también se define como V(G)=P+1, donde P es el número de nodos predicado (nodos de donde parten al menos dos aristas) contenidos en el grafo de flujo G.

La complejidad ciclomática del código se calculó en sus tres variantes.

Sustituyendo:

1. El grafo de flujo tiene 2 regiones, por tanto, V(G) = 2
2. V(G) = 6 - 6 + 2 = 2

3. V(G) = 1 + 1 = 2

Por tanto, la complejidad ciclomática del grafo de flujo es 2.

3) Determinar un conjunto básico de rutas linealmente independientes:

Después de calcular la complejidad del grafo, se pudo comprobar que los resultados obtenidos son iguales a 2; los conjuntos de caminos básicos son:

- Camino básico 1: I-1-2-3-F
- Camino básico 2: I-1-2-4-F

4) Obtención de casos de prueba (CP):

Una vez definidas las rutas, se procede a diseñar los casos de prueba para cada una de las rutas básicas obtenidas. A continuación, se muestra el caso de prueba para el camino básico 2:

Tabla 4. Caso de prueba camino básico 2

Descripción	El botón muestra las opciones de ajustes si se selecciona la opción con valor 1 o muestra la opción de información de la aplicación si se selecciona el valor con número 2.
Condición de ejecución	Número de valor seleccionado.
Entradas	El botón de ajuste y el botón acerca de.
Resultados esperados	Se realiza la navegación a la opción de información de la aplicación

Fuente: Elaboración propia.

Al aplicar el caso de prueba expuesto anteriormente se evidenció que el flujo de trabajo de las funcionalidades es correcto, cumpliéndose las condiciones de la prueba y el resultado esperado es satisfactorio.

3.2.2 Pruebas a nivel de sistema

Se preparara un ambiente de pruebas, el cual debe ser similar al que tiene el usuario para que las pruebas sean efectivas y demuestren los bugs que el usuario podría encontrar mientras realiza sus actividades diarias (Paredes, 2018).

Método de Caja Negra: Las pruebas de caja negra consisten en probar un sistema o programa informático sin tener conocimiento previo de su funcionamiento interno. Esto no sólo se refiere

a no conocer el código fuente en sí, sino que implica no haber visto ninguna de las documentaciones de diseño que rodean al software (Sánchez, 2015).

Para la aplicación de este método se utiliza la Técnica de partición de equivalencia que a continuación se describe.

Técnica de partición de equivalencia: divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El método se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada (Pressman, 2010).

Para emplear esta técnica se realizan los diseños de casos de pruebas los cuales son una evaluación de las clases de equivalencia para una condición de entrada.

Como resultado de aplicar la técnica partición equivalente se realizaron tres iteraciones de pruebas, para poder alcanzar resultados satisfactorios, atendiendo al correcto comportamiento del sistema ante diferentes situaciones y obteniéndose los resultados que se muestran en la siguiente figura:

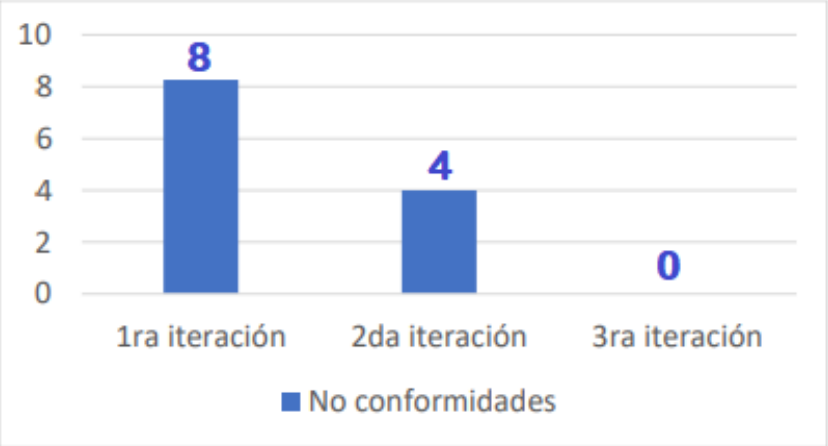


Figura 13. Representación de la cantidad de NC por iteraciones

Fuente: Elaboración propia

Estas no conformidades (NC) se clasificaron en: Validación, Interfaz y Ortografía. En las dos primeras iteraciones la tendencia de NC fue de Interfaz y Ortografía. En cada una se corrigieron las NC dando paso a que en la tercera iteración no se detectaran NC.

3.3 Valoración de la viabilidad de la propuesta de solución

Para valorar la viabilidad de la propuesta de solución se aplicó una encuesta a especialistas de la UCI y el MINJUS arrojando un resultado positivo. A continuación, se muestra una tabla donde se evalúa un antes y un después de la solución a partir de las principales deficiencias identificadas en la situación problemática y de los resultados de la encuesta (Anexo. 2) realizada, con el propósito de verificar cómo, mediante la aplicación móvil desarrollada, si contribuye a facilitar la consulta de información actualizada sobre las notarías en Cuba.

Tabla 5. Valoración de la viabilidad de la propuesta de solución

Antes	**Después**
Las secciones marco legal, directorio de unidades y de unidades notariales.) se encuentran desactualizadas, provocando molestias en las entidades notariales correspondientes y confusiones en los usuarios a la hora de realizar los trámites que en la aplicación se mencionan.	Con la realización de esta aplicación se logró un alto nivel de satisfacción del cliente, el 80 % de los encuestados (8) califica la primera impresión de la APK como muy buena y el 10% (1) como buena y el 10% (1) como no tan buena, lo que evidencia que la solución satisface las necesidades del cliente. La solución favorece una mayor disponibilidad de la información notarial en el país, sin necesidad de internet para su consulta, contando con mecanismos para facilitar la consulta de una forma interactiva y comprensible para los usuarios. La aplicación constituye otra vía para la consulta de información notarial en el país incrementando el acceso a la información de la misma.
La aplicación móvil no permite el ajuste de la apariencia; dígase colores, tamaño del texto y tema. Elemento que limita a las personas con dificultades visuales a acceder al contenido.	

Fuente: Elaboración propia.

Conclusiones del capítulo

- Con la aplicación de las técnicas de validación de requisitos, se logró ratificar que los mismos definen la aplicación móvil que el cliente desea.

- La ejecución de las pruebas de software establecidas por la metodología AUP variación UCI, en las disciplinas de Pruebas internas a nivel de unidad y de sistema, aplicando las pruebas funcionales, mediante el método caja negra, y las pruebas unitarias mediante el método caja blanca, comprobó el correcto funcionamiento de la propuesta de solución y que esta se acoge a las expectativas del cliente.
- Por último, el análisis del comportamiento de la variable de investigación, demostró que el desarrollo de la versión 2.0 de iLex Notario contribuye a facilitar la consulta de información notarial en el país.

CONCLUSIONES FINALES

Al finalizar la presente investigación para el desarrollo de la versión 2.0 de la aplicación móvil iLex Notario se pudo llegar a las siguientes conclusiones:

- El estudio y el análisis de los principales referentes teóricos-metodológicos en los que se sustenta la presente investigación, permitió elaborar el marco teórico, facilitando la adquisición de los conocimientos necesarios para la solución propuesta.
- La especificación de requisitos, el análisis y diseño de la propuesta de solución permitió generar los artefactos y la obtención de los elementos necesarios para un adecuado y estructurado proceso de implementación.
- Los resultados obtenidos a través de la aplicación de la métrica calidad de especificación y las pruebas de software, así como la valoración de la viabilidad de la propuesta de solución, comprobaron de forma cuantitativa la calidad de la versión 2.0 de la aplicación móvil iLex Notario.

REFERENCIAS BIBLIOGRÁFICAS

Abrutsky, A. (2016). *Implantación de un sistema de control de versiones*. https://openaccess.uoc.edu/handle/10609/55246

Alarcón, A., & Sandoval, E. (2008). Herramientas CASE para ingeniería de Requisitos. *Cultura Científica*. https://revista.jdc.edu.co/index.php/Cult_cient/article/view/305

Alfonso, D. (2017). *Herramienta para generar productos de trabajos de la metodología variación AUP-UCI*. https://repositorio.uci.cu/jspui/handle/123456789/8141

Andrés, R. (2017). Métodos de Investigación—Concepto, función y ejemplos. *Concepto*. https://concepto.de/metodos-de-investigacion/

Araneda, C. E. (2015). La función pública notarial y la seguridad jurídica respecto de la contratación electrónica en el Perú. *Universidad Privada Antenor Orrego - UPAO*. https://repositorio.upao.edu.pe/handle/20.500.12759/1223

Arias, J. F. (2022). *Repercusión de arquitectura limpia y la norma ISO/IEC 25010 en la mantenibilidad de aplicativos Android*. http://www.scielo.org.co/scielo.php?pid=S0123-77992021000300226&script=sci_arttext#f3

Bellas, F. G., Unanue, R. M., & Fernández, V. D. F. (2016). *Lenguajes de programación y procesadores*. https://books.google.com.cu/books?id=eHL-DAAAQBAJ&printsec=frontcover&hl=es#v=onepage&q&f=false

Benítez, R. (2016). La función notarial. *Materiales de Estudio*. https://publicaciones.fder.edu.uy/index.php/me/article/view/38

Boisselier, L. M. (2021). *Sistema clasificador de emociones para las comunicaciones digitales privadas por medio de técnicas de inteligencia artificial* [bachelorThesis]. https://repositorio.uesiglo21.edu.ar/handle/ues21/22207

Castellanos Rodríguez, H. H., & Osorio Franco, J. F. (2018). *Módulo para la evaluación de estándares de codificación bajo la metodología de calidad de software para la universidad de cundinamarca* [Thesis]. https://repositorio.ucundinamarca.edu.co/handle/20.500.12558/1085

Dart. (2023a). *Dart overview*. https://dart.dev/overview.html

Dart. (2023b). *Effective Dart: Style*. https://dart.dev/effective-dart/style.html

Díaz, I. (2021). *Ilex Asesoramiento Jurídico: La nueva aplicación móvil del MINJUS*. https://www.directoriocubano.info/acontecer/ilex-asesoramiento-juridico-la-nueva-aplicacion-movil-del-minjus/

Díaz-Canel, M. (2018). *Díaz-Canel: La informatización de la sociedad es clave para la participación ciudadana y la soberanía | Cubadebate*. http://www.cubadebate.cu/noticias/2018/12/18/diaz-canel-la-informatizacion-de-la-sociedad-es-clave-para-la-participacion-ciudadana-y-la-soberania/

Eguía Peña, B. (2002). *El desarrollo de las tecnologías de la información y la comunicación: Un nuevo reto para el mercado de trabajo.* https://dialnet.unirioja.es/servlet/articulo?codigo=631529

Estévez, J. R. (2020). La ingeniería de requisitos en el desarrollo de aplicaciones informáticas. *Revista Cubana de Informática Médica*, *12*(2), Article 2.

Flutter. (2023). *Flutter—Build apps for any screen*. //flutter.dev/

Fontela, C. (2012). *UML: Modelado de Software para Profesionales*. Alpha Editorial. https://books.google.es/books?hl=es&lr=&id=6gZzEAAAQBAJ&oi=fnd&pg=PP1&dq=UML&ots=FB8Ra59gyA&sig=SQbi1C9gIvbDKRTvESjwdDTPhyA#v=onepage&q=UML&f=false

García, I. C., & Mesa, M. L. C. (2019). Las generaciones digitales y las aplicaciones móviles como refuerzo educativo. *Revista Metropolitana de Ciencias Aplicadas*, *2*(1), Article 1.

GitLab. (2023). *The DevSecOps Platform*. https://about.gitlab.com/

González, J. (2021, junio). *Diseño e implementación de un SDK Android para facilitar la interacción de aplicaciones móviles con una blockchain* [Info:eu-repo/semantics/bachelorThesis]. E.T.S. de Ingenieros Informáticos (UPM). https://oa.upm.es/68118/

Guerrero, C. A., Suárez, J. M., & Gutiérrez, L. E. (2013). Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*, *24*(3), 103-114. https://doi.org/10.4067/S0718-07642013000300012

Hernández, L. R. B., Peña, D. M., Valdés, O. R., & Cornelio, O. M. (2016). Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso. *Serie Científica de la Universidad de las Ciencias Informáticas*, *9*(7), Article 7. https://publicaciones.uci.cu/index.php/serie/article/view/822

Jiménez, L. (2023). *▷ Todo sobre dispositivos móviles: Definición y características*. https://leojimzdev.com/todo-sobre-dispositivos-moviles-definicion-y-caracteristicas/

JSON. (2023). *JSON*. https://www.json.org/json-en.html

Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software*. https://repositorio.uca.edu.ar/handle/123456789/522

Martínez, J. C., Henao, C., Henao, F., & Zapata, E. (2021). Utilización de Arquitecturas Limpias para Trabajo con Buenas Prácticas en la Construcción de Aplicaciones Java. *Revista*

Innovación Digital y Desarrollo Sostenible - IDS, *1*(2), Article 2. https://doi.org/10.47185/27113760.v1n2.37

Menzinsky, A., López, G., Palacio, J., Sobrino, M. Á., Álvarez, R., & Rivas, V. (2022). *Historias de Usuario. Ingeniería de requisitos ágil*. https://www.scrummanager.com/files/scrum_manager_historias_usuario.pdf

MINCOM. (2018). *DECRETO-LEY No. 370 "SOBRE LA INFORMATIZACIÓN DE LA SOCIEDAD EN CUBA"*. https://www.mincom.gob.cu/sites/default/files/marcoregulatorio/dl_370-18_informatizacion_sociedad.pdf

MINJUS. (2023). *Misión y funciones*. MINJUS. https://www.minjus.gob.cu/es/mision-y-funciones

Monterrubio, E. (2019). Sistema Operativo. *Con-Ciencia Serrana Boletín Científico de la Escuela Preparatoria Ixtlahuaco*, *1*(1), Article 1. https://repository.uaeh.edu.mx/revistas/index.php/ixtlahuaco/article/view/3672

Paredes, D. M. (2018). Proceso de pruebas del sistema del fondo de empleados usando ISTQB – marzo 2018. *Universidad Nacional Mayor de San Marcos*. https://cybertesis.unmsm.edu.pe/handle/20.500.12672/11195

Patterson, A. D. (2015). *Hacia los IDE de siguiente generación: Revisión de herramientas de desarrollo*. https://repositorio.ulacit.ac.cr/handle/123456789/4781

Pentón, F. A. M., Fernández, Y. A., Pérez, Y. G. D., Gallo, J. M. F., & Pérez, J. C. A. (2021). Proyecto LexCuba: Gobierno electrónico en su móvil. *Revista Cubana de Transformación Digital*, Article 2. https://rctd.uic.cu/rctd/article/view/42

Pressman, R. S. (2010). *Ingenieria del Software. Un Enfoque Practico*. http://artemisa.unicauca.edu.co/~cardila/IS__Libro_Pressman_7.pdf

Ramírez, P. (2022, enero 4). *¿Cuáles son los sistemas operativos más usados en 2022?* https://itsoftware.com.co/content/sistemas-operativos-mas-usados/

Roa, J., Gutiérrez, M., & Stegmayer, G. (2008). FAIA: Framework para la enseñanza de agentes en IA. *IE comunicaciones : revista iberoamericana de informática educativa*. https://redined.educacion.gob.es/xmlui/handle/11162/5926

Rodríguez Peña, A. D., & Silva Rojas, L. G. (2016). Arquitectura de software para el sistema de visualización médica Vismedic. *Revista Cubana de Informática Médica*. http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S1684-18592016000100006&lng=es&nrm=iso&tlng=es

Sánchez, J. M. (2015, junio 16). *Pruebas de Software. Fundamentos y Técnicas* [Info:eu-repo/semantics/bachelorThesis]. E.T.S.I y Sistemas de Telecomunicación (UPM). https://oa.upm.es/40012/

Sommerville, I. (2011). *Ingeniería de Software*. https://gc.scalahed.com/recursos/files/r161r/w25469w/ingdelsoftwarelibro9_compressed.pdf

Tabares, R. B. (2010). Patrones Grasp y Anti-Patrones: Un Enfoque Orientado a Objetos desde Lógica de Programación. *Entre Ciencia e Ingeniería*, *4*(8), Article 8.

Visual Studio Code. (2023). *Documentation for Visual Studio Code*. https://code.visualstudio.com/docs

ANEXOS

ANEXO 1. Entrevista.

Guía de preguntas utilizadas en el desarrollo de la entrevista a los profesionales del proyecto LexCuba y directivos del MINJUS.

1. ¿Qué personas proveerán la información, ¿Cuál es su grado de autoridad y que disponibilidad de tiempo tienen?
2. ¿Qué funciones desean que tenga la aplicación móvil?
3. ¿Qué objetivo tiene el producto?
4. ¿A qué usuarios va dirigido el producto?
5. ¿Podrá el usuario adaptar o personalizar el contenido?
6. ¿Qué información se quiere comunicar?
7. ¿Qué nivel de accesibilidad a la información tendrán los usuarios finales?
8. ¿Poseen documentación sobre la normativa?
9. ¿Poseen alguna aplicación semejante al producto que desean? ¿Cuál?
10. En caso de ser positiva la pregunta anterior especificar cuáles son las deficiencias de la aplicación móvil existente.
11. ¿En qué infraestructura tecnológica se va a desplegar el producto?
12. ¿Cuáles son los actores fundamentales para el funcionamiento del proceso?
13. ¿Cómo funcionan los procesos que se desean informatizar?
14. ¿Cuáles son los roles que intervienen en el proceso?

ANEXO 2. Encuesta para evaluar la satisfacción con la aplicación móvil iLex Notario 2.0.

Según su criterio, responda a las siguientes preguntas:

1- ¿Como describiría su primera impresión sobre nuestra aplicación?

__Muy buena __Buena __No tan buena

2- Del 1 al 5, donde uno es malo y 5 es excelente, ¿cómo ha sido su experiencia con nuestra aplicación?

__1 __2 __3 __4 __5

3- ¿Cuántas veces ha utilizado la aplicación?

__Más de 2 __ Más de 10

4- ¿La calidad del producto es la esperada?

__Si __ No

5- ¿Considera que iLex Notario 2.0, satisface sus necesidades en cuanto al diseño e información que brinda?

__Supera mis expectativas.

__Resuelve mis necesidades sin más.

__Es útil, pero hay mejores opciones.

__No resuelve mis necesidades.

ANEXO 3. Diseño de caso de prueba del RF17 “Modificar Tema”.

Figura 14. DCP, “Modificar Tema”

Descripción general

El caso de prueba inicia cuando el usuario presiona el botón "Cambiar Tema" en la sección de ajustes de la aplicación.

Condiciones de ejecución

SC Modificar Tema

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Modificar tema	*Permite cambiar el tema de la aplicación a un color oscuro.*	*V* *N/A*	*Muestra toda la interfaz visual de la aplicación con un tema de color oscuro.*	*Ajustes/Cambiar Tema*
EC 1.2 Modificar tema oscuro	*Permite cambiar el tema de la aplicación a un color claro.*	*V* *N/A*	*Muestra toda la interfaz visual de la aplicación con un tema de color claro.*	*Ajustes/Cambiar Tema*